

What is claimed is:

1 1. A method comprising:

2 allocating a memory entry in a memory device to
3 instructions executed on a multithreaded engine included
4 in a packet processor, a portion of the memory entry
5 includes a unique identifier assigned to the
6 instructions.

1 2. The method of claim 1, further comprising:

2 maintaining a count of threads included in the
3 multithreaded engine that use the memory entry.

1 3. The method of claim 1, further comprising:

2 maintaining a bit to represent availability of the
3 memory entry for thread use.

1 4. The method of claim 2 wherein maintaining the count

2 includes incrementing the count to represent a thread
3 initiating use of the memory entry.

1 5. The method of claim 2 wherein maintaining the count

2 includes decrementing the count to represent a thread halting
3 use of the memory entry.

1 6. The method of claim 3 wherein maintaining the bit
2 includes setting the bit to represent availability of the
3 memory entry for thread use.

1 7. The method of claim 3 wherein maintaining the bit
2 includes clearing the bit to represent unavailability of the
3 memory entry for thread use.

1 8. The method of claim 3, further comprising:
2 checking the bit to determine the availability of
3 the memory entry for thread use.

1 9. The method of claim 1 wherein the unique identifier
2 includes four bits.

1 10. The method of claim 1 wherein the memory entry identifies
2 a location in a local memory included in the multithreaded
3 engine of the packet processor.

1 11. A computer program product, tangibly embodied in an
2 information carrier, the computer program product being
3 operable to cause a machine to:

4 allocate a memory entry in a memory device to
5 instructions executed on a multithreaded engine included
6 in a packet processor, a portion of the memory entry

7 includes a unique identifier assigned to the
8 instructions.

1 12. The computer program product of claim 11 being further
2 operable to cause a machine to:

3 maintain a count of threads included in the
4 multithreaded engine that use the memory entry.

1 13. The computer program product of claim 11 being further
2 operable to cause a machine to:

3 maintain a bit to represent availability of the
4 memory entry for thread use.

1 14. The computer program product of claim 12 wherein
2 maintaining the count includes incrementing the count to
3 represent a thread initiating use of the memory entry.

1 15. The computer program product of claim 12 wherein
2 maintaining the count includes decrementing the count to
3 represent a thread halting use of the memory entry.

1 16. The computer program product of claim 13 wherein
2 maintaining the bit includes setting the bit to represent
3 availability of the memory entry for thread use.

1 17. The computer program product of claim 13 wherein
2 maintaining the bit includes clearing the bit to represent
3 unavailability of the memory entry for thread use.

1 18. The computer program product of claim 13 being further
2 operable to cause a machine to:

3 check the bit to determine the availability of the
4 memory entry for thread use.

1 19. The computer program product of claim 11 wherein the
2 unique identifier includes four bits.

1 20. The computer program product of claim 11 wherein the
2 memory entry identifies a location in a local memory included
3 in the multithreaded engine of the packet processor.

1 21. A memory manager comprises:

2 a process to allocate a memory entry in a memory
3 device to instructions executed on a multithreaded engine
4 included in a packet processor, a portion of the memory
5 entry includes a unique identifier assigned to the
6 instructions.

1 22. The memory manager of claim 21, further comprises:

2 a process to maintain a count of threads included in
3 the multithreaded engine that use the memory entry.

1 23. The memory manager of claim 21, further comprises:

2 a process to maintain a bit to represent
3 availability of the memory entry for thread use.

1 24. A system comprising:

2 a packet processor capable of,
3 allocating a memory entry in a memory device to
4 instructions executed on a multithreaded engine
5 included in a packet processor, a portion of the
6 memory entry includes a unique identifier assigned
7 to the instructions.

1 25. The system of claim 24 wherein the packet processor is
2 further capable of:

3 maintaining a count of threads included in the
4 multithreaded engine that use the memory entry.

1 26. The system of claim 24 wherein the network processor is
2 further capable of:

3 maintaining a bit to represent availability of the
4 memory entry for thread use.

1 27. A network forwarding device comprising:

2 an input port for receiving packets;
3 an output for delivering the received packets; and
4 a network processor capable of,

5 allocating a memory entry in a memory device to
6 instructions executed on a multithreaded engine
7 included in a packet processor, a portion of the
8 memory entry includes a unique identifier assigned
9 to the instructions.

1 28. The network forwarding device of claim 27, wherein the
2 network processor is further capable of maintaining a count of
3 threads included in the multithreaded engine that use the
4 memory entry.

1 29. The network forwarding device of claim 28, wherein the
2 network processor is further capable of maintaining a bit to
3 represent availability of the memory entry for thread use.

1 30. A method comprising:

2 allocating a content-addressable-memory (CAM) entry
3 to a microblock executed on a multithreaded microengine
4 included in a network processor, a portion of the CAM
5 entry includes a unique identifier assigned to the
6 microblock.

1 31. The method of claim 30, further comprising:

2 maintaining a count of threads included in the
3 multithreaded microengine that use the CAM entry.

1 32. The method of claim 30, further comprising:

2 maintaining a bit in a status register to represent
3 availability of the CAM entry to identify a local memory
4 location.